Technická univerzita v Liberci

FAKULTA PŘÍRODOVĚDNĚ-HUMANITNÍ A PEDAGOGICKÁ

Katedra: Ústav nových technologií a aplikované informatiky

Studijní program: Specializace v pedagogice

Studijní obor (kombinace): Informatika a Geografie

APLIKACE PRO SBĚR DAT S URČENOU ZEMĚPISNOU POLOHOU

SOFTWARE APPLICATION FOR COLLECTING DATA WITH GEOSPATIAL INFORMATION

Bakalářská práce: 10-FP-NTI-03

Autor: **Podpis:**

Michal Těhník

Adresa:

Roosweltova 9

46851, Smržovka

Vedoucí práce: Mgr. Jiří Vraný Ph.D.

Konzultant: Ing. Miloš Turek

Počet

stran	grafů	obrázků	tabulek	pramenů	příloh
59	0	20	0	12	2

V Liberci dne: 23. 07. 2010

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

V Liberci dne: 23. 04. 2010 Michal Těhník

Poděkování

Rád bych poděkoval panu Ing. Miloši Turkovi, že mi vyšel vstříc a umožnil mi psát o vlastním tématu. Stejně tak mu děkuji za vstřícnost a odborné vedení mé práce.

Anotace

Tato práce pojednává o vývoji aplikace pro sběr dat s určenou geografickou polohou. Aplikace je určená pro mobilní platformu Windows Mobile. Dále obsahuje porovnání ostatních mobilních platforem a navigačních systémů. Dalšími částmi práce jsou úvod do problematiky navigace na planetě Zemi, možnosti vývoje aplikací pro mobilní platformu od Microsoftu a popis prostředků k tomu určených. Hlavním cílem je vyvinout aplikaci pro Windows Mobile. Jednotlivým stěžejním bodům se věnují samostatné kapitoly. Jako jsou využité návrhové vzory nebo testování aplikace.

Klíčová slova: prostorová data, programování, Windows Mobile, Microsoft

Annotation

This work concerns itself with the development of a software application for collecting data with given geospatial position. The application is meant for the Windows Mobile operating system. This work also contains a comparison of other mobile platforms and navigation systems. The rest of the work introduces problems concerning the navigation on planet Earth, the possibilities of an application development for the Microsoft's mobile platform, and a description of tools and resources meant for such an application development. The main goal of this work is to develop a software application for Windows Mobile however. Each main topic of this work, like the layout patterns used in the application or the testing of the application, is presented in a separate chapter.

Key words: geospatial data, programming, Windows Mobile, Microsoft

Annotation

Die Arbeit behandelt die Entwicklung der Anwendungssoftware für Sammel der Daten mit bestimmter geographischen Lage. Die Anwendungssoftware ist für die bewegliche Plattform Windows Mobile bestimmt. Der Vergleich weiterer beweglichen Plattformen und navigatorischer Systeme wird noch umfasst. Im Rest werden die Problematik der Navigation auf der Erde, die Möglichkeiten der Entwicklung der Anwendungssoftware für beweglicher Microsoftplattform und Beschreibung der Mittel dafür bestimmt angeführt. Der Zweck dieser Arbeit ist die Anwendungssoftware für Windows Mobile zu entwickeln. Selbständige Kapitel widmen sich einzelnen essentialen Punkten wie z.B. benutzten Aufrissmustern oder Testierung der Anwendungssoftware.

Schlüsselwörter: die Geodaten, die Programmierung, Windows Mobile, Microsoft

Obsah

1 Úvod

Cenová dostupnost přijímačům GPS signálu způsobila značné rozšíření těchto zařízení mezi veřejnost. Jedná se zejména o ruční přístroje pro určování polohy, které jsou určené pro turistiku nebo autonavigace využívající GPS. Společně s tímto jevem se v poslední době klade i vysoký důraz na spojení různých jevů, událostí nebo objektů s jejich polohou. Díky tomuto jsme schopni jednodušeji modelovat různé situace nebo výskyt určitých jevů lépe než tomu bylo dříve, kdy bylo nutné polohu vyhledat na mapě. Dnes, pokud známe souřadnice, můžeme přesně určit jejich polohu na již existující mapě. Dalším dílkem do této skládačky je významné rozšíření internetových mapových služeb. Tomuto odvětví jednoznačně vévodí společnost Google se svým portálem Google Maps. Které lze například využít pro vizualizaci dat získaných pomocí ručního přístroje GPS, určeného pro turistiku a zobrazit si například poslední trasu.

Významným milníkem byl vznik zařízení, které kombinovalo mobilní telefon s operačním systémem, ten zajišťoval platformu pro vývoj aplikací, a přijímačem GPS signálu. Mobilní telefon zde pak může hrát roli jako poskytovatel internetové konektivity, která se využívá pro odesílání dat z aplikace. Díky tomuto je možné přímo v terénu pracovat s aplikacemi, které využívají určování polohy. Ať již se jedná o jednoduché aplikace, které pouze odesílají údaje o aktuální poloze na vzdálený server nebo aplikace aktivně komunikující s dalšími podobnými zařízeními. Zde stojí za zmínku aplikace Google Latitude, která umožňuje odesílat informace o aktuální poloze a zobrazovat ji veřejně na internetu jiným uživatelům. Na druhé straně pak stojí aplikace určené pro práci, jako je například ESRI ArcGIS Mobile, který poskytuje základní vlastnosti GIS systémů i na mobilní zařízení.

Právě možnost práce v terénu je důležitou výsadou těchto zařízení. Se zlepšující se hardwarovou vybaveností zařízení, jak po stránce výkonu zařízení, tak i přibývaní různých čidel, se způsoby využití rozšiřují. Díky možnosti spojit získaná data v terénu s jejich přesnou polohou nám poskytuje nový rozměr chápání těchto informací.

2 Vymezení tématu

Práce se zabývá vývojem aplikace pro sběr dat s určenou zeměpisnou polohou, která je určena pro mobilní zařízení s operačním systémem Windows Mobile. Jako platforma se používá prostředí .NET Compact Framework, které umožňuje vytváření aplikací v managed prostředí. Jako úložiště dat je využito Microsoft SQL Server Compact. Zdrojový kód aplikace je v jazyce C#.

2.1 Motivace

Téma vývoje aplikace pro sběr dat s určenou zeměpisnou polohou jsem zvolil, z několika důvodů. Prvním je můj zájem o oblast mobilních zařízení, zvláště pak ta, která využívají operační systém Windows Mobile. Pro vývoj aplikace jsem vybral platformu .NET Compact Framework, která nabízí moderní principy návrhu a vývoje aplikací, a zároveň poskytuje rychlý a snadný způsob vývoje softwaru ve spojení s vývojovým prostředím Visual Studio 2008.

Existence této aplikace by výrazně usnadnila práci všem, kteří potřebují sbírat nějaká data v terénu, znát jejich přesnou polohu a tyto informace následně zpracovávat. Ať již jsou důvody pro tuto činnost profesionální, amatérské nebo školní.

Jednoduchým příkladem může být seminární práce studenta geografie na téma „Brownfields v Libereckém kraji". Brownfields jsou opuštěné budovy nebo plochy, které z důvodu nevyužívání chátrají. Zpravidla se jedná o staré průmyslové objekty. Tato práce zahrnuje mapování takovýchto objektů, včetně jejich charakteristiky, stavu a polohy. Díky této aplikaci student může velice jednoduše v terénu zmapovat potřebné údaje a poté je jednoduše zpracovat do své práce. Usnadněna mu je zejména vizualizace do mapy a práce se získanými daty.

2.2 Očekávané vlastní přínosy

Přínosem této práce je aplikace, která umožní sběr dat v terénu a jejich pozdější zpracování. Dále je to seznámení s problematikou návrhu a vývoje aplikace pro platformu Windows Mobile postavenou nad .NET Compact Frameworkem.

3 Geodata

Jako geodata lze označit veškerá data, která sebou nesou i informaci o poloze. Ať už se jedná o polohu, kde byla pořízena nebo data hodnota je sama o sobě údajem. Tyto údaje se vztahují k určitým místům v prostoru. Většinou se jedná o polohu na planetě Zemi, ale není to podmínkou. Stejně tak jako využíváme například polohu o různých místech na Zemi, můžeme tohoto využít i v menším systému, jako je například sklad. Kde budeme uchovávat informaci o tom, kde se nachází dané zboží a místo geodetických souřadnic můžeme využít číslo regálu a číslo příslušné sekce v regálu.

O datech, která využívají pro určení polohy geografických souřadnic, hovoříme jako o georeferencovaných. Tyto informace je možné jednoznačně umístit do prostoru. Zdroje pro informaci o poloze dělíme na primární a sekundární.

Primární zdroje jsou takové, které zahrnují získávání přímo v terénu. Jedná se o mapování povrchu, terénní průzkum, dálkový průzkum země a v našem případě zejména globální družicový polohový systém pro získávání geodetické polohy na zemském tělese.

Sekundárním zdrojem jsou informace, které jsou získané z primárních zdrojů, resp. z jichž existujících. Lze do nich zařadit papírové a digitální mapy, statistiky, ale také data z dálkového průzkumu země nebo geoinformační systémy.

Prostorová data jsou nejčastěji prezentována v podobě map, ať už analogových – tištěných nebo digitálních. Papírové mapy plní funkci prezentační, tak archivační. Pro digitální prostorová data se používá jako úložiště databáze nebo speciální formáty pro ukládání prostorových dat, například ESRI shapefile.

Základem pro prostorová data je zejména to, že věci se nacházejí někde v prostoru, události se stávají na nějakém určitém místě. Většina lidských činností je vázána na určité místo na zemském povrchu. Díky propojení informací a událostí s místem jejich výskytu můžeme zjišťovat různé statistické údaje nebo modelovat vývoj určitých procesů. Například díky informací z katastrálního úřadu dokážou systémy GIS určit nejvhodnější polohu pro výstavbu benzínové pumpy, kterou nelze postavit kdekoliv, ale dané místo musí splňovat několik parametrů, jako je dostupnost vody, stabilní podloží nebo určitá vzdálenost od zdrojů pitné vody. [5]

4 Souřadnicové systémy

Pro využívání georeferencovaných dat je nutná existence souřadnicového systému. K tomuto systému jsou vztaženy jednotlivé body, resp. souřadnice těchto bodů. Pro vyjádření polohy na Zemi je využito geodetického systému. Samotné definování tohoto systému je složitá úloha, protože planeta Země není kulatá, ale jedná se o geoid. Toto označení reprezentuje fyzikální model povrchu, který nejpřesněji vystihuje tvar Země, kdy povrch klidné mořské hladiny, by pokračoval myšleně pod kontinenty. Povrch Geoidu je ve všech bodech kolmý na tížnici (směr tíhové síly). Směr tížnic ovlivňuje nepravidelné rozmístění hmoty v zemské kůře. Povrch geoidu je zvlněný i pod kontinenty i oceány. Díky tomuto není vhodný jako výpočetní plocha pro definici geodetického systému. [5]

Náhradou za geoid se používá obecný zemský geoid, jenž je matematicky definovaný. Střed Země souhlasí se středem geoidu, delší poloosa leží v rovině rovníku a kratší poloosa je shodná s rotační osou Země. Toto matematické vyjádření Země se postupně zlepšovalo, proto existuje několik typů elipsoidů (Bessel, Krassovsky, WGS60, WGS-84).

Díky moderní technice a vesmírným družicím bylo do roku 1984 získáno dostatečné množství informací pro výpočet a definování rotačního elipsoidu. Tento elipsoid nese označení WGS-84.

Každý stát využívá souřadnicový systém, který nejlépe vyhovuje danému území státu. V České republice se historicky využívalo více souřadných systémů, které se od sebe lišily. Toto bylo způsobeno jako důsledek různých období a politických poměrů v dobách, kdy se dané systémy vytvářely. Dalším kritériem byla kvalita trigonometrické sítě. V českých zemích je nejvyužívanější systém S-JTSK pro civilní sektor a S-42 pro sektor vojenský, který je nyní nahrazen systémem WGS-84.

Systém GPS využívá geodetický systém WGS-84, který využívá armáda Spojených států Amerických a je zároveň standardem NATO. Díky vstupu České republiky je i tento systém standardem také armády ČR, konkrétně od 1. ledna 2006.

4.1 WGS-84

World Geodetic System 1984 je celosvětově uznávaný standard. Jedná se o geodetický geocentrický systém armády USA. V tomto standardu také pracuje globální družicový systém pro určování polohy NAVSTAR-GPS a je také standardizován jako geodetický systém NATO.

Světový geodetický systém 1984 je konvenční terestrický systém, který má základ v modifikaci družicového navigačního systému Transit, který byl provozován námořnictvem Spojených států amerických. Tato modifikace spočívá v posunu počátku souřadnicového systému, změny rotace a měřítka tak, aby systém byl geocentrický a referenční nultý poledník byl identický se základním poledníkem.

Systém je definován jako geodetický geocentrický systém s pravotočivou kartézskou soustavu souřadnic se středem v těžišti Země. Osa x směřuje k průsečíku nultého poledníku a rovníku, osa z k severnímu pólu a osa y je na obě předchozí kolmá.

WGS-84 používá zeměpisné souřadnice. Jednotlivé body jsou definovány zeměpisnou délkou, šířkou a výškou. Zápis je ve stupních, ve stupních a minutách nebo ve stupních, minutách a vteřinách (s jednotkami v desetinném tvaru). Nutné je vždy uvést zda se jedná o severní nebo jižní šířku a o východní nebo západní délku. Výška se uvádí v metrech, nejedná se ve skutečnosti o skutečnou nadmořskou výšku, ale o elipsoidickou, která se velice blíží té skutečné.

Česká armáda, jako součást NATO, využívá světový geodetický systém WGS-84 od 1. ledna 2006. Původní systém S-42 se tímto přestal používat. S tím souvisí i nahrazení původních topografických metod a technologií, technologiemi využívajících tento systém, což je systém GPS. [11]

5 Globální družicový polohový systém

Určení polohy na planetě Zemi je odvěkým problémem. Pro každou civilizaci v historii naší planety hrála roli poloha a mapování svého okolí, což dokládá i nejstarší kartografická památka, která byla nalezena u Pavlova na jižní Moravě v roce 1962. Jedná se o mamutí kel, na kterém je vyobrazeno okolí tábořiště. Tato památka se datuje do mladšího paleolitu, tedy cca. 28 000 let před naším letopočtem.

Dalším přelomovým objevem v určování polohy byl kompas, který využívá magnetický pól Země. První zařízení tohoto druhu se datují do 4. století na území Číny.

Průlom v určování polohy na zemském povrchu přísel s dobýváním vesmírů. Díky družicím tento obor dostal zcela novou perspektivu. První s využitím družicového systému byli Spojené státy americké, které s vypouštěním družic začali v roce 1959. Jednalo se o systém Transit, který v dnešní době již není v provozu. V tehdejším bipolárním světě Rusko nechtělo zůstat pozadu a v roce 1967 začalo s výstavbou vlastního systému Parus, který v různých modifikacích existoval až do roku 1995. [11]

V dnešní době existuje několik projektů globálních družicových polohových systémů, ale jediným plně aktivním je americký systém Global Positioning System. NAVSTAR-GPS, což je jeho oficiální název. Tento systém je provozovaný Ministerstvem obrany Spojených států amerických. Konkurenčními systémy jsou GLONASS, který je pod správou Ruska a během roku 2010 by mělo dojít k jeho plnému zprovoznění, tak aby pokrýval celou zeměkouli. Dále je to projekt Evropské unie Galileo, který měl být dle původního plánu v provozu od roku 2010, nový plán však hovoří o roku 2014. Čínským konkurentem je systém Compass, který je také ve fázi výstavby, dle plánu by měl během letošního roku pokrývat území Číny a do roku 2020 se stát globálním.

Dále existují systémy, které pokrývají pouze část území. Je jím Beidou 1, který je předchůdcem čínského globálního systému Compass. Druhým systémem, který je aktivně v provozu je DORIS, který provozuje Francouzská vesmírná agentura. Dále existují dva systémy, které jsou ve výstavbě a to je Indický IRNSS a Japonský QZSS.

V příloze B je uveden přehled družicových polohových systémů.

5.1 NAVSTAR-GPS

Systém, známý pod zkratkou GPS, je vojenský globální družicový polohový systém, který je provozovaný Ministerstvem obrany Spojených států amerických. Mezi jeho hlavní služby patří určení polohy a přesného času kdekoliv na planetě Zemi nebo nad ní. [4]

5.1.1 Historie

Projekt je následovníkem původního amerického systému Transit, který byl aktivní v letech 1964 – 1996. Vývoj NAVSTAR-GPS počal sloučením projektů amerického letectva pro určování polohy a systému pro určování přesného času, který byl pod záštitou amerického námořnictva. Testovací provoz probíhal v letech 1973 – 1979. Během tohoto období probíhala výběrová řízení na dodavatele vesmírné části systému a pozemních uživatelských přijímačů. Raketa s první družicí byla na oběžnou dráhu vynesena roku 1978. Druhá fáze započala v roce 1979 a spočívala ve vybudování hlavního řídicího střediska a vynesení 28 družic do vesmíru, tato fáze skončila roku 1985. Náplní třetí fáze bylo nahrazení starých družic, novějšími druhé generace. Dne 3. 3. 1994 byl systém prohlášen za schopný provozu. Systém NAVSTAR-GPS se skládá ze tří segmentů. Vesmírný, řídicí a uživatelský.

5.1.2 Vesmírný segment

Je tvořen družicemi, které obíhají kolem Země ve vzdálenosti 20 183 km nad povrchem Země. Celkem je ve vesmíru 30 družic rozdělených na 6 oběžných drah. Dráhy jsou od sebe v 60° intervalech. Jedna družice obsahuje mimo komunikačních a energetických systémů, také 3 až 4 atomové hodiny a optické, rentgenové a pulzní-elektromagnetické senzory pro detekci startů balistických raket a jaderných výbuchů.

5.1.3 Řídící segment

Jedná se o část systému, která se stará o monitoring a údržbu vesmírného segmentu. Mezi úkoly tohoto segmentu patří úprava oběžných drah jednotlivých družic, kalibrace atomových hodin. Velitelství pozemní části se nachází na letecké základně Los Angeles, řídicí středisko je na letecké základně v Colorado Springs. Existuje ještě záložní středisko v Geithersburgu ve státě Maryland. Dále jsou zde 3 povelová stanice a 18 stanic monitorovacích.

Kdyby došlo ke zničení pozemního segmentu, jsou jednotlivé družice schopny autonomního provozu, který zajišťuje životnost systému dalších 6 měsíců. Tento režim umožňuje komunikaci jednotlivých družic mezi sebou.

5.1.4 Uživatelský segment

Jedná se o samotné pasivní přístroje pro příjem GPS signálu z družic, které jsou v danou chvíli pro zařízení viditelné. Na základě přijatých informací, tato zařízení vypočítávají polohu, nadmořskou výšku a mohou zobrazit přesný čas a datum. GPS přijímače jsou pouze pasivní, tudíž družice mohou obsloužit jejich neomezený počet. Dále lze uživatelský segment rozdělit na dvě skupiny.

Autorizovaní uživatelé, což je armáda USA, mají k dispozici klíče, které dekódují signál s vyšší přesností. Toto se využívá zejména na podporu vojáků v poli, vojenskou geodézii a vyšší přesnost času. Například střela s plochou dráhou letu Tomahawk využívá k navádění na cíl systém GPS.

Ostatní uživatelé, zjednodušeně řečeno se jedná o civilní sektor. Zde je využití prakticky neomezené, i když s menší přesností, než mají autorizovaní uživatelé. Využívá se v dopravě, geografii, turistice nebo zábavě.

Jelikož systém NAVSTAR-GPS je jediným systémem pokrývajícím celou planetu Zemi, tak jediná široce dostupná zařízení umějí pracovat pouze s tímto systémem. Právě toto platí zejména v mobilních zařízeních, které se vyrábějí pouze s přijímačem systému GPS.

6 Platforma

V dnešní době existuje několik mobilních platforem, které poskytují možnost vyvíjet vlastní aplikace. Pro tuto práci jsem vybral Windows Mobile, která poskytuje vývoj pod .NET Compact Frameworkem, což je podmnožina .NET Frameworku známého ze stolní verze Windows. Dále Windows Mobile poskytují kvalitní databázové prostředí SQL Server Compact, který je odnoží Microsoft SQL Serveru.

Nelze říci, která platforma je nejlepší, každá má své pro a proti. Pro samotného vývojáře hrají významnou roli zejména předešlé zkušenosti. Java vývojář zřejmě sáhne po Androidu. Tato platforma pochází z dílny společnosti Google, který určuje její další vývoj. Programátor se zkušenostmi z .NET Frameworku například sáhne právě po Windows Mobile, které umožňují běh jeho podmnožiny.

Na druhou stranu i platformy, které nejsou příbuzné s nějakou velkou vývojovou platformou, mohou získat významný podíl na trhu mobilních zařízení. Právě tímto je iPhone od společnosti Apple. Tento systém byl uveden na trh v roce 2007 a těší se nebývalé popularitě i přes naprostou uzavřenost systému a horší podporu vývojářů oproti ostatním platformám.

Dalším faktorem pro vývojáře je i kvalita vývojového a testovacího prostředí pro daný operační systém. V tomto bezesporu opět vede Microsoft s vývojovým prostředím Visual Studio, které podporuje vývoj do předešle verze 2008, stávající verze 2010, která byla uvedena na jaře roku 2010, neposkytuje podporu pro zařízení Windows Mobile. Na srovnatelné úrovni je i systém Android, který má IDE postavené na vývojovém prostředí Eclipse, ale existují i pluginy do dalších prostředí určené pro Javu, jako jsou NetBeans nebo IntelliJ Idea. Shodou okolností obě původně od českých firem.

V příloze A je uveden stručný přehled stávajících mobilních platforem a stručný popis možností vývoje pro ně.

7 Windows Mobile

7.1 Historie

Vývoj Windows Mobile, resp. Windows CE, jsou datovány do roku 1992. Microsoft měl cíl přinést na trh operační systém pro mobilní zařízení. Bohužel, v této době ještě neexistoval dostatečně výkonný hardware pro tak smělé cíle, proto byl projekt ukončen nezdarem v roce 1994. V následujícím roce 1995 se začalo s testováním nového druhu mobilního zařízení s operačním systémem ušitým na míru na architektuře procesoru MIPS 3000 a MIPS 4000. Tímto se datuje vznik Windows CE 1.0.

Windows CE tvoří základ pro Windows Mobile, jehož první verze byla vydána v roce 2000 pod názvem Pocket PC 2000. V následující verzi 2002 došlo k rozdělení platformy na tři edice. Pocket PC 2002 pro zařízení typu PDA, verze Pocket PC 2002 Phone Edition, pro zařízení která obsahovala GSM modul a třetí verze byla pojmenována Smartphone 2002. Další verze 2003 přinesla podporu WiFi a Bluetooth.

Postupem času došlo k přejmenování na Windows Mobile, jehož poslední významná verze nese číslo 6.0. Tato verze přinesla několik významných vlastností jako je podpora VoIP, rozšíření možností Bluetooth, podporu AJAXu, JavaScriptu v mobilní verzi Internet Exploreru.

Poslední vydanou verzí je Windows Mobile 6.5, která byla uvedena na trh na podzim roku 2009. Její přínosy jsou pouze kosmetické, zahrnují úpravu “Today” obrazovky, vylepšení odezvy systému nebo podporu dotykový gest. [20]

Na konferenci MIX 2010, která se každoročně koná v Las Vegas, Microsoft uvedl zbrusu nový systém pro mobilní telefony, který je stále postaven na Windows CE, ale využívá nové technologie Microsoftu, hlavně Silverlight pro vytváření uživatelského prostředí. Tento systém je zaměřen na spotřebitele. Stávající systém Windows Mobile byl přesunut pod oddělení starající se o Windows Embedded, pod které spadají i Windows CE, tudíž i Windows Mobile.

7.2 Architektura

Windows Mobile jsou postavené nad operačním systémem Windows CE, který je otevřeným 32-bitový systém s multitaskingem a multithreadingem pro různá mobilní zařízení jako jsou PDA, smartphony, prodejní terminály a další zařízení. Návrh uživatelského rozhraní měl za cíl co největší podobnost s Windows pro počítače.

Systém obsahuje podporu internetových protokolu, rozhraní Win32 API pro programování aplikací. Samotný OS se skládá z modulů, které mohou být skládána pro potřeby různých implementací pro různá hardwarová zařízení. Windows CE mají podporu USB zařízení, tisku, paměťových médii, ale také velké škály procesoru: Intel X-Scale, Hitachi SH4, Samsung nebo NEC.

7.2.1 Kernel

Jádro systému je vícevláknové s preemptivním multitaskingem. Je založeno na platformě Win32. Existuje zde několik úrovní priorit vláken procesů. Má podporu stránkování pamětí RAM a ROM. Díky OAL jádro pracuje na více typech procesorů.

7.2.2 Hardware

Díky jisté benevolentnosti a otevřenosti Windows Mobile je lze nalézt na různých druzích zařízení, počínaje PDA a Smartphony, pro které jsou určena. Také je lze nalézt na různých typech tabletPC, případně na některých průmyslových zařízeních.

Samotné zařízení je většinou doplněno o různé hardwarové komponenty a senzory. Mezi ně patří digitální kamera, WiFi, Bluetooth, GPS modul, GSM modul, FM rádio, IrDA, gravitační senzor nebo digitální kompas. Samotný hardware je většinou postaven na mikroprocesorech ARM, které jsou 32bitový s architekturou RISC.

Důležitým rozdělením pro svět Windows Mobile je to jestli zařízení má dotykový displej nebo nemá. Zařízení s dotykovým displejem většinou nemají další hardwarová tlačítka, zůstávají pouze nejnutnější ovládací prvky, jako jsou tlačítka pro příjem a odmítnutí hovoru, případně nastavení hlasitosti. Zařízení bez dotykového displeje označována jako Smartphone disponují klasickou telefonní klávesnicí známou z běžných mobilních telefonů.

8 Možnosti vývoje aplikací

Microsoft poskytuje vlastní nástroje a prostředky pro vývoj, testování a nasazení aplikací pro Windows Mobile. Díky přímé podpoře Microsoftu je vývoj značně usnadněn. Existují tři základní cesty jakými je možno se vydat, pokud se rozhodneme vyvíjet aplikaci pro mobilní zařízení. [19]

8.1.1 Nativní kód

Vývoj aplikací v nativním kódu je umožněn v jazyce C++, konkrétně v Microsoft Visual C++. Jedná se o rozšíření jazyka C, poskytující základní koncepty objektově orientovaného jazyka, jako jsou objekty nebo dědičnost. Všechny potřebné nástroje jsou poskytovány v rámci vývojového prostředí Microsoft Visual Studio a Windows Mobile SDK. Existují i volně dostupné kompilátory a vývojová prostředí pro vývoj mobilních aplikací. Výhodou je možnost vyvíjet i na jiných operačních systémech než Windows, jako je Linux nebo Mac OS X.

Vývoj nativního kódu je vhodný zejména pro aplikace, kde je vyžadovaný vysoký výkon aplikace a její nízká paměťová náročnost. Dalšími výhodami je přímé využití systémových knihoven operačního systému, nízkoúrovňový přístup k hardwaru zařízení nebo možnost využití stávajícího kódu z jiné C++ aplikace. Například kód, který obstarává nějakou business logiku aplikace.

Každý vývoj nativní aplikace klade vyšší nároky na vývojáře, protože se musí v kódu starat zejména o alokaci a uvolňování zdrojů, které využívá jeho aplikace. Nativní kód se využívá pro ovladače zařízení, rozšiřování systému, jako je modifikace “Today” obrazovky nebo změna uživatelského prostředí. Dalším využitím jsou například hry, kde je požadován co největší výkon a zároveň nízká náročnost na daný hardware.

8.2 Řízené prostředí

Existuje několik možností jak vyvíjet řízený kód pro zařízení se systémem Windows Mobile. Je možné využít Java Micro Edition, což je Java určená pro mobilní zařízení. JavaME je doma hlavně na zařízeních, které neobsahují mobilní operační systém, což je způsobeno omezeními ze strany Java Virtual Machine určené pro tyto zařízení. Tato práce se zabývá vývojem pro .NET Compact Framework, který bude popsán blíže v následující kapitole.

Výhod tohoto prostředí je několik. Tato mezivrstva se stará o běh aplikace, o přiřazování a uvolňování zdrojů, které jsou potřeba. Stejně tak poskytuje lepší zabezpečení a spolehlivost než nativní aplikace. Jsou to záležitosti, které programátor aplikace nemusí řešit, protože se o ně stará nižší vrstva běhového prostředí.

Aplikace napsané pro virtuální prostředí se při kompilaci, překládají do mezijazyka. Tento mezikrok je využit pro generování cílového strojového kódu na konkrétním virtuálním stroji a počítač, resp. zařízení. Toto umožňuje JIT (Just In Time) kompilaci, kdy je program přeložen ve chvíli spuštění, přímo do strojového kódu počítače. Výhodou je lepší optimalizace a využití speciálních instrukcí, které poskytují různé typy procesorů.

Tato mezivrstva poskytuje možnost vývoje aplikací v různých programovacích jazycích, které jsou posléze přeloženy do mezijazyka a využívají společné knihovny a prostředí. Díky tomuto může vývojář zvolit optimální druh jazyka, který se nejlépe hodí pro řešení daného problému.

Existují dva velcí zástupci tohoto přístupu. Prvním je Java Virtual Machine, která využívá Java Bytecode a jak název napovídá je určena pro běh programů napsaných v Jave. Dalšími podporovanými jazyky jsou například JavaScript, Python nebo Ruby. Druhým zástupcem je .NET Framework a jeho opensource alternativa Mono, tyto prostředí podporují Common Language Infrastructure. Zde se využívá Common Intermediate Language, který je mezi jazykem pro toto prostředí. .NET Framework díky tomuto poskytuje podporu pro Python, Ruby nebo Lisp.

8.3 Webové aplikace

V dnešní době je jasným trendem vytváření webových aplikací, které běží přímo ve webovém prohlížeči. Jedná se zejména o tzv. RIA (Rich Internet Application), které se snaží smazávat rozdíl mezi aplikacemi určenými pro běh na počítači a těmi co běží v prohlížeči.

Hlavním rozdílem webových aplikací určených pro mobilní zařízení a těmi běžnými je zejména cílové zařízení, na kterém se budou zobrazovat. Je potřeba brát ohled na možnosti internetového připojení takového zařízení, které je v dnešní době v drtivé většině realizováno přes mobilní sítě GSM, proto je potřeba brát zřetel na velikost stránky ve smyslu přenášených dat. Druhým zásadním omezením je velikost displeje, na kterém bude aplikace zobrazena. Jedná se o velikost v řádu jednotek palců, ty největší zařízení v tomto segmentu mají velikost okolo 4 palců. Toto přináší dva požadavky pro tvůrce aplikace. Prvním je, aby texty v aplikaci byly dobře čitelné na malém displeji i z větší vzdálenosti. V drtivé většině se v dnešní době tato zařízení ovládají prsty, proto je třeba i ovládací prvky stránky uzpůsobit tomuto požadavku. Velikost prvků by měla odpovídat velikosti konečku prstu a zajišťovat tím bezproblémové ovládání bez použití dotykového pera nebo jiného prostředku pro ovládání zařízení.

Výhodou těchto aplikací je jejich správa. Jsou umístěné na vzdáleném serveru v prostředí internetu a existuje pouze jedna instance této webové aplikace. Díky tomuto se vývojář, popřípadě administrátor nemusí starat o problematiku nasazení a aktualizace této aplikace napříč zařízeními, které ji používají. Všechno je umístěno na serveru, kde stačí aktualizovat aplikaci a v tu chvíli je aktuální verze dostupná všem jejím uživatelům.

Nevýhodou těchto aplikací je zejména potřeba permanentního připojení k internetu, resp. ke vzdálenému serveru, což přináší zvýšenou spotřebu energie z baterií. Další překážkou může být omezené možnosti využití hardwarových komponent mobilního zařízení, jako je GPS modul nebo digitální kompas. Nevýhodou je i nemožnost ukládat data lokálně na zařízení. Všechna tato omezení se snaží odstranit specifikace HTML5, která ještě není hotová, ale přináší možnost práce s GPS pozicí, lokální úložiště dat nebo podporu lokální SQLite databáze.

9 Nástroje

9.1 .NET Compact Framework

Jedná se o moderní, objektově orientované a snadno použitelné prostředí pro vývoj aplikací. Lze ho rozdělit na dvě nedílné součásti a to knihovnu funkcí a běhové prostředí.

.NET Framework je knihovna, která je značně rozsáhlá, poskytuje zobrazení dialogů, volání základních služeb operačního systému, poskytuje metody pro práci s databází nebo zprostředkovává webové služby.

Dále je to běhové prostředí, ve kterých běží programy. Jedná se o aplikace, které jsou napsány pro platformu .NET. Toto prostředí zajišťuje správu spuštěných vláken, práci s pamětí nebo garbage collector.

Díky tomu, že jsou aplikace při kompilaci přeloženy do mezi-kódu CIL a následně pomocí JIT kompilátoru zkompilovány na daném prostředí, je zde možnost výběru širokého množství programovacích jazyků. Hlavním jazykem .NET Frameworku je C# a Visual Basic.NET, dalšími jsou F# (funkcionální programovací jazyk) nebo dvojice dynamických jazyků IronPython a IronRuby.

.NET Compact Framework je podmnožinou .NET Frameworku a je určen pro mobilní zařízení, konkrétně pro operační systém Windows CE, na kterém jsou Windows Mobile postaveny. Mimo knihoven, které mají původ v .NET Frameworku, obsahuje i speciální knihovny, které jsou určeny pro mobilní zařízení, jako je například práce s GSM, GPS nebo obsluha dotykových obrazovek.

První verze .NET Compact Frameworku byla uvedena na trh na konci roku 2002 a snaží se kopírovat novinky, které přináší jak .NET Framework určený pro operační systém Windows, tak hardwarové novinky na poli mobilních zařízení. Poslední vydání s číslem verze 3.5 spatřilo světlo světa počátkem roku 2008. Tato verze přináší podporu WCF (Windows Communication Foundation), což je rozhraní pro programování service-oriented aplikací. Dalším významnou součástí je podpora LINQ (Language-Integrated Query). Jedná se o jazyk z dílny Microsoftu, který přináší nový způsob pro dotazování nad jakýmikoliv daty, ať už se jedná o databázi, XML nebo kolekcemi objektů. [10]

9.2 SQL Server Compact Edition

Jedná se o jednu z edic Microsoft SQL Server, která je zaměřena pro využití v jednodušších aplikacích, které nepotřebují funkce velkého robustního serverového řešení. Jednou z cílových platforem pro tuto verzi jsou i mobilní zařízení.

Hlavním rozdílem mezi Compact Edition a ostatními verzemi, počínaje verzí Express, je ten, že databázový engine neběží jako služba na pozadí. Engine běží ve stejném procesu jako samotná aplikace, která využívá jeho prostředky pro práci s daty. Databáze je zde reprezentována pouze jediným souborem s příponou SDF.

Tato edice je zejména určena pro menší aplikace, které prací s počty záznamů v řádech tisíců a desetitisíců. Dalšími omezeními je nepřítomnost procedur a triggerů. Tato edice nabízí podporu transakcí, omezení referenční integrity nebo podporu více připojení k jedné databázi.

S SQL Server Compact Edition se z prostředí .NET Compact Frameworku možnost práce pomocí ADO.NET, což je API pro práci s daty. Na toto navazuje možnost Entity Frameworku, což je ORM nástroj obsažený v .NET Frameworku. Dále podporuje i dotazovací jazyk LINQ. [14]

Pro mobilní zařízení existují i další databázové servery, jako je SQL Anywhere od společnosti Sybase nebo open source databáze SQLite, která je podobná SQL Serveru Compact, také je tvořena jedním souborem a engine běží v rámci aplikace, která využívá databázi.

9.3 Visual Studio

Je IDE (Integrated Development Environment) od společnosti Microsoft. Nejedná se o vývojové prostředí určené pro jeden konkrétní jazyk, ale je to spíše platforma, která podporuje různé jazyky, pomocí rozšíření.

Prostředí má podporu code-complete pod názvem IntelliSense, které pomáhá programátorovi při psaní kódu. Jsou zde možnosti pro refaktoring, podporu verzovacích systémů a další.

Visual Studio je distribuováno v několika placených edicích a ve verzi Express je zadarmo pro komerční účely. Podporovaný je bezpočet jazyků, počínaje C#, přes C++, F#, JavaScript, konče u Pythonu a Ruby. Poslední verzí je Visual Studio 2010.

Já jsem pro vývoj aplikace využíval verzi Visual Studio 2008 Professional Edition, která podporuje vývoj pro mobilní zařízení.

9.4 Jazyk C#

Je programovací jazyk z dílny Microsoftu, určený pro .NET Framework, je standardizovaný jako ECMA-334 a ISO/IEC 23270. Jedná se o nepřímého následovníka jazyka C, kterému je podobný zejména svou syntaxí. A také je přímím konkurentem Javy od společnosti SUN Microsystems, dnes již Oracle.

Hlavním architektem jazyka je Anders Hejlsberg, který je autorem Turbo Pascalu a byl také vedoucím tvůrcem vývojového prostředí Delphi.

C# je moderní objektově orientovaný programovací jazyk. Mezi jeho hlavní rysy patří jednoduchá dědičnost a možnost násobné implementace rozhraní. Dále automatická správa paměti a využití garbage collectoru. V neposlední řadě to jsou zpracování chyb pomocí výjimek, typová bezpečnost a atributové programování. [13]

Vývojový tým se snaží implementovat moderní vlastnosti, jako je podpora generik (parametrizované typy), částečných tříd (partial class), Lambda výrazů nebo LINQ.

10 Aplikace

10.1 Analýza

Aplikace pro sběr dat s určenou zeměpisnou polohou by měla splňovat následující:

Možnost využívání aplikace více uživateli. Mohou nastat situace, kdy je zařízení sdíleno, například v rámci týmu, oddělení nebo skupinou studentů. Dále mohou být informace o uživatelích využity v dalších verzích aplikace, které mohou využívat připojení na vzdálený server přes internet.

Definovat projekty. Jelikož využití aplikace je zaměřené na sběr dat pro jednotlivé úkoly, měla by aplikace umožňovat definování a správu jednotlivých projektů. Jako projekt lze například chápat různé seminární práce nebo mapování v různých městech, resp. územích. Každý uživatel by měl mít k dispozici pouze projekty, které jím byly vytvořeny.

Možnost definovat si vlastní skupinu dat v rámci projektu, která bude pro každý bod společná. Hlavním využitím aplikace je sběr dat, tudíž by měl mít uživatel možnost definovat si, jaké informace chce sbírat v rámci daného projektu. Zároveň by zde měla být i možnost definovat jakého typu jsou jednotlivé položky v této skupině.

Jak napovídá název aplikace, je předpoklad, že data o poloze budou získávána z GPS modulu. Jelikož existuje pouze jediná služba poskytující informace o poloze, kterou je právě americký systém GPS, nelze v této oblasti vybírat. Také zařízení s Windows Mobile je schopno spolupracovat pouze s tímto systémem, resp. s hardwarovými moduly GPS.

Jelikož pro získávání údajů o poloze bude využito systému GPS, měla by aplikace umožňovat zobrazení informací z tohoto systému. Jako je počet viditelných družic, aktuální poloha nebo přesný čas.

Export dat by měl být nedílnou součástí aplikace, aby bylo možno s nimi pracovat po jejich sběru. Zde se týká hlavně využití v různých GIS softwarech, jako je například ESRI ArcGIS nebo v tabulkovém procesoru.

11 Architektura

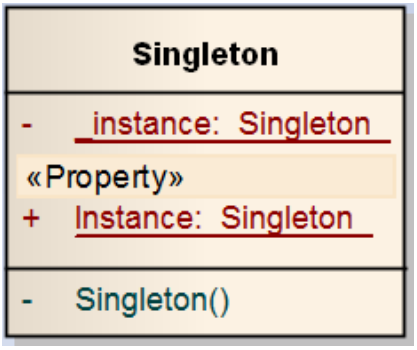
11.1 Návrhové vzory

Jsou nedílnou součástí softwarového vývoje. Návrhové vzory reprezentují obecné řešení problému, který se často vyskytuje při programování aplikací. Jejich původ není v softwarovém inženýrství, ale jsou běžnou součástí mnoha jiných a starších oborů, takovým příkladem může být architektura.

11.1.1 Singleton

Je jedním z nejjednodušších návrhových vzorů, do češtiny lze název přeložit jako jedináček. Řeší problém, kdy potřebujeme zajistit, aby v aplikaci existovala pouze jediná instance určité třídy. [6]

Typickým příkladem využití je třída obsluhující práci s databází, kde potřebujeme zabezpečit to, že nebude existovat nadměrný počet spojení do databáze.



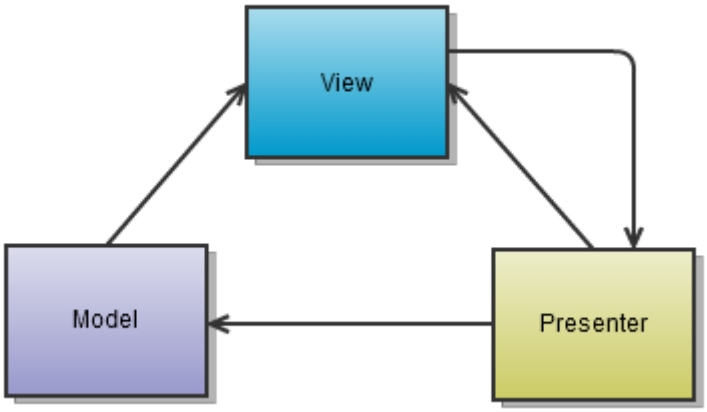
Obrázek 11-1 – UML digram implementace Singletonu

```
public class DataLayerImpl : IDataLayer
{
    private static DataLayerImpl _instance;

    public static DataLayerImpl Instance
    {
        get
        {
            return _instance ?? (_instance = new DataLayerImpl());
        }
    }
}
```

11.1.2 Model-View-Presenter (MVP)

Je návrhový vzor, který je odvozen od Model-View-Controler (MVC), který získává stále na popularitě. Vděčíme za to hlavně webovému vývoji, ať již to jsou Ruby on Rails, Zend Framework pro PHP nebo ASP.NET MVC od Microsoftu. Všechny tyto frameworky se snaží stavět na vzoru MVC. Tento návrhový vzor dělí samotnou aplikaci nebo její komponenty na tři logické části, tak aby byly na sobě co nejvíce nezávislé a změny v jedné neovlivnily ostatní dvě. Tyto části jsou Model, View a Controller. Model jako takový reprezentuje data a business logiku, View se stará o zobrazení uživatelského rozhraní a Controller zajišťuje obsluhu událostí v aplikaci a samotnou aplikační logiku.



Obrázek 11-2 – Schéma vzoru Model-View-Presenter

Model-View-Presenter je znám hlavně na poli desktopových aplikací nebo RIA technologií, jako jsou Flex, Silverlight nebo JavaFX, dále se využívá v ASP.NET Web Forms. Rozdílů oproti samotnému MVC je několik. Model zůstává zachován a obstarává data a business logiku. View se stará navíc o obsluhu uživatelských událostí, což může být například kliknutí na tlačítko. Ovšem tato událost se pouze deleguje zavoláním metody na Presenteru. Mít ve View jakoukoliv aplikační logiku je chyba. Presenter, který zastupuje Controller v původním MVC, se stará o aplikační a prezentační logiku, pracuje s Modelem a zajišťuje aktualizaci View. [7]

MVP využívám pro zobrazení jednotlivých obrazovek uživatelského rozhraní, které jsou tvořeny UserControly a jsou umístěny na hlavním formuláři aplikace, který zde působí jako kontejner.

11.1.3 Objektově-relační mapování (ORM)

Je technika, která je zajišťuje konverzi dat mezi relační databází a objekty v objektově orientovaných programovacích jazycích. Výsledkem je ukládání dat v relační databázi a objektový model dat na aplikační úrovni. De facto se jedná o perzistenci mezi těmito modely reprezentace dat. Největší přínos ORM je práce přímo s objekty místo SQL příkazů.

Využití ORM výrazně zjednodušuje vývoj aplikace, zrychluje ho a zároveň umožňuje je její snadnou rozšiřitelnost. Samotné mapování mezi objekty a tabulkami je jednoduše provedeno jako přiřazení vlastností objektů jejich ekvivalentním sloupcům v tabulce.

Ve světě běžných aplikací, resp. aplikací desktopových a webových existují desítky nástrojů pro objektově-relační mapování. Ze světa Javy nelze nezmínit nástroje, jako jsou JPA (Java Persistence API) nebo Hibernate. Na platformě Windows jsou to například ADO.NET Entity Framework nebo ekvivalent Javovského Hibernate, čímž je Nhibernate. Všechny zmíněné nástroje jsou k dispozici zdarma.

Tyto nástroje poskytují široké možnosti v oblasti ORM, jako je synchronizace mezi objektovým a databázovým modelem. Místo toho, aby programátor definoval oba modely a staral se o jejich synchronizaci z pohledu návrhu, tak obojí definuje na jednom místě a ORM nástroj se již postará o vygenerování obou modelů. Dalšími prvky jsou například cachování dat pro omezení databázových dotazů nebo lazy-loading, který zajišťuje načtení dat až ve chvíli, kdy jsou potřeba.

Bohužel pro prostředí Windows Mobile, resp. .NET Compact Framework neexistuje podobně široká nabídka nástrojů. Existují pouze placené nástroje EntitySpaces (300USD za vývojářskou licenci), LLBLGen Pro (299EUR za jednu licenci) nebo eXpressPersistent Objects Suite od společnosti Devexpress (149,99USD za jednu licenci). Existuje open-source projekt pro ORM na platformě .NET Compact Frameworku a to SubSonic, sice se vývojáři na stránkách projektu chlubí podporou Microsoft SQL Server CE, ale tato podpora nebyla dosud implementována. Za viníka nedostupnosti těchto nástrojů lze považovat omezené možnosti .NET CF.

12 Objektový model

Na diagramu tříd, který je zobrazen na obrázku 12-1, je patrné, že pro reprezentaci dat v aplikaci jsou využity objekty typu POCO (Plain Old CLR Object). Tento typ je charakteristický tím, že samotné třídy nemají žádné metody a jedná se pouze o nositele dat v rámci aplikace.

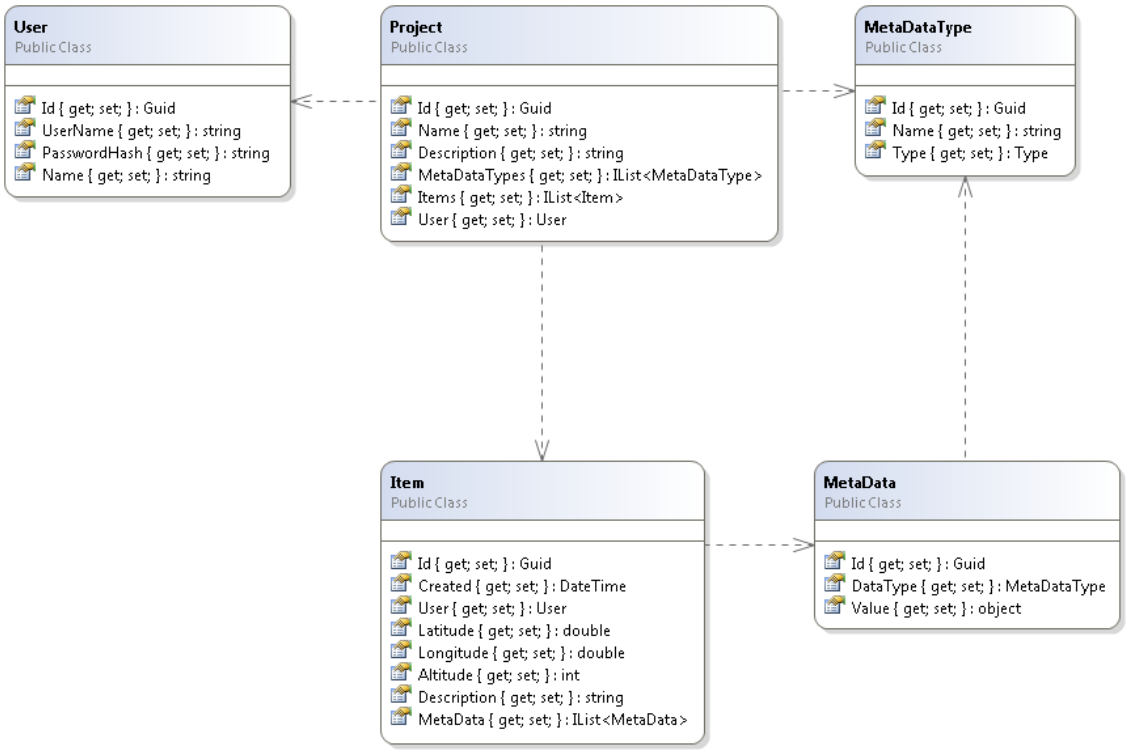
Třída User zde reprezentuje uživatele v rámci aplikace, která podporuje více uživatelů, kteří mohou pracovat s aplikací. Jedná se pouze o identifikaci uživatele a jeho autentifikaci pomocí hesla, ve chvíli, kdy se přihlašuje do aplikace.

Třída Project reprezentuje projekt, v rámci kterého se sbírají veškeré údaje, které chce uživatel zaznamenávat. Tato třída obsahuje kolekci objektů Item a MetaDataType. Poslední zmíněný je zde uveden jako seznam sledovaných atributů v rámci projektu a na základě tohoto se generují formuláře pro zadání dat k jednotlivým objektům Item.

Třída Item je každý jednotlivý záznam s geografickou polohou, která obsahuje potřebné údaje, jako zeměpisnou šířku a délku, dále informaci o nadmořské výšce. K této třídě jsou přirazena kolekce objektů MetaData.

Třída MetaData je určena pro vlastní data, která jsou sbírány v rámci projektu. Property Value je typu object z důvodu, že hodnota může být různých datových typů, aplikace nyní podporuje typy string, integer, float a boolean.

Třída MetaDataType reprezentuje uživatelem definované datové atributy, které je možno sbírat v rámci projektu ke kterému je možno je přiřadit.

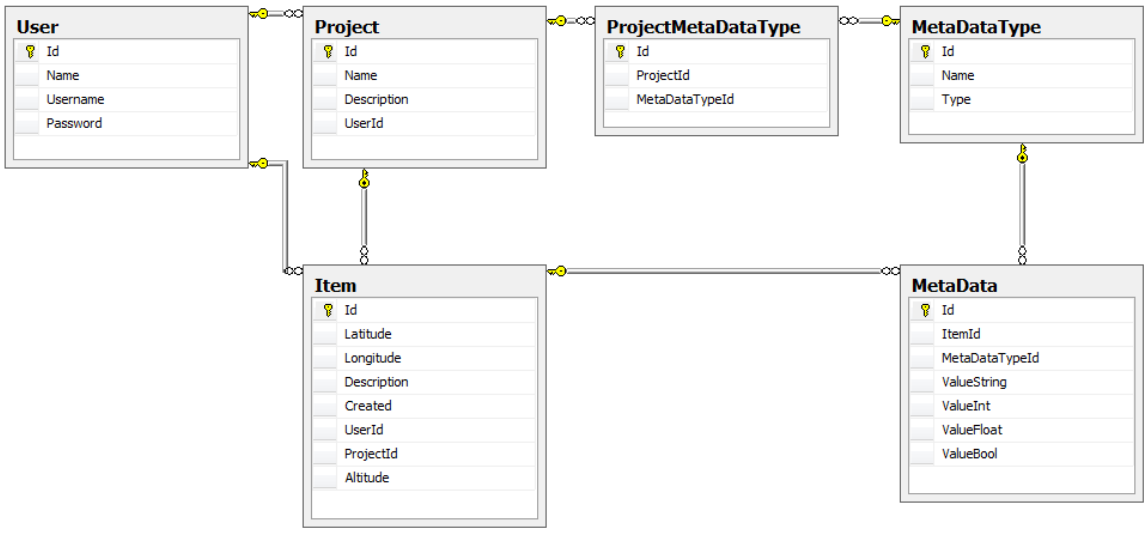


Obrázek 12-1 – Diagram tříd

13 Databázový diagram

Je zobrazením fyzické struktury databáze, resp. tabulek a jejich datových atributů. Jednotlivé tabulky jsou úložištěm pro výše zmíněné objekty z objektového modelu aplikace. Je využito Microsoft SQL Server Compact Edition, tudíž se jedná o relační databázi. Tabulky obsahují primární klíče jako identifikátory pomocí datového typu uniqueidentifier. Dále jsou tabulky propojeny cizími klíči pro zajištění integrity dat.

Za zmínku stojí vazební tabulka ProjectMetaDataType, díky které je možné využívat MetaDataType napříč projekty a tudíž nedochází k duplicitám definovaných typů v rámci celé aplikace.



Obrázek 13-1 – Diagram databáze

14 Omezení platformy

Jak jsem již zmiňoval, mobilní platformy sebou přinášejí jistá omezení, se kterými se musí programátor vypořádat. Já jsem se při programování aplikace setkal s dvojicí významných překážek, které jsem se snažil řešit svým vlastním přístupem.

14.1 App.config

Windows Mobile nepodporují soubor App.Config, kde je uloženo výchozí nastavení aplikace. Z toho důvodu jsem se rozhodl vytvořit si vlastní implementaci načítání tohoto nastavení ze souboru App.Config.

```
public static NameValueCollection LoadConfig()
{
    string appPath =
    Path.GetDirectoryName(Assembly.GetExecutingAssembly().GetName().CodeBase);
    string configFile = Path.Combine(appPath, "App.config");

    if (File.Exists(configFile) == false)
    {
        return null;
    }

    XmlDocument oXml = new XmlDocument();

    oXml.Load(configFile);

    XmlNodeList oList = oXml.GetElementsByTagName("appSettings");

    NameValueCollection appSettings = new NameValueCollection();

    foreach (XmlNode oNode in oList)
    {
        foreach (XmlNode oKey in oNode.ChildNodes)
        {
            appSettings.Add(oKey.Attributes["key"].Value,
            oKey.Attributes["value"].Value);
        }
    }

    return appSettings;
}
```

14.2 Objektově-relační mapper

Jak jsem zmínil v kapitole 11.1.3, tak pro .NET Compact Framework neexistují žádné volně dostupné ORM frameworky, které by zajišťovali perzistenci mezi objektovým a databázovým modelem. Proto jsem naimplementoval vlastní zajištění tohoto mapování.

Moje implementace se skládá z několika metod, které zajišťují zmíněné O/R mapování.

Metoda ExecuteNonQuery, která přijímá parametrem SqlCeCommand, který reprezentuje samotný dotaz do databáze. V tomto případě metoda nemá žádnou návratovou hodnotu, protože je určená pro dotazy INSERT (vložení záznamu), UPDATE (změna záznamu) a DELETE (odstranění záznamu). Důležité je po každém vykonání dotazu uzavřít připojení do databáze, aby nezastávala zbytečné otevřená a případně neblokovala další spojení.

```
protected void ExecuteNonQuery(SqlCeCommand command)
{
    using (SqlCeConnection connection = new SqlCeConnection(_connectionString))
    {
        connection.Open();
        command.Connection = connection;

        using (command)
        {
            command.ExecuteNonQuery();
        }

        connection.Close();
    }
}
```

Ukázka metody pracující s ExecuteNonQuery může být ItemRemove, která odstraňuje z databáze záznam o objektu Item. Z předaného objektu se získá jeho Id, které je reprezentováno typem Guid. Toto Id se do Sql dotazu předá pomocí parametru, což zajišťuje obranu proti Sql Injection.

```
public void ItemRemove(Item item)
{
    SqlCeCommand command = new SqlCeCommand("DELETE Item WHERE Id = @Id");
    command.Parameters.AddWithValue("@Id", item.Id);
    ExecuteNonQuery(command);
}
```

Dále je to metoda ExecuteReader, která pracuje také přímo s databází. Parametry jsou SqlCeCommand, jako v předešlé metodě, a ReadDataHandler, který je předáván formou delegátu SqlCeDataReader, který vrací jednotlivé řádky výsledku na daný dotaz.

Jak již bylo zmíněno, tato metoda se stará o dotaz SELECT (načtení záznamu), která vrací jednotlivé záznamy, resp. řádky z databáze.

```
protected void ExecuteReader(SqlCeCommand command,
                             ReadDataHandler readDataHandler)
{
    using (SqlCeConnection connection = new SqlCeConnection(_connectionString))
    {
        connection.Open();
        command.Connection = connection;

        using (command)
        {
            using (SqlCeDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    readDataHandler(reader);
                }
            }
        }

        connection.Close();
    }
}

protected delegate void ReadDataHandler(SqlCeDataReader reader);
```

V obou metodách pro práci s databází se využívá konstrukce using, která zajišťuje, že po vykonání kódu je nad daným objektem zavolána metoda Dispose pro uvolnění prostředků. Ovšem se nejedná o nic jiného než o přehlednější zápis try-finally bloku. Zvýrazněnou část převede kompilátor na následující kód.

```
SqlCeDataReader reader = command.ExecuteReader();

try
{
    while(reader.Read())
    {
        readDataHandler(reader);
    }
}

finally
{
    reader.Dispose();
}
```

Další metodou je Reader, konkrétně ProjectReader, který vrací generickou indexovanou kolekci objektů IList. Ve výše zmíněném delegátu se zde vytváří pro každý vrácený řádek, resp. záznam z databáze nový objekt a jednotlivými hodnotami v řádku se naplní prvky nového objektu.

```
private IList<Project> ProjectReader(SqlCeCommand command)
{
    IList<Project> projects = new List<Project>();

    ExecuteReader(command, delegate(SqlCeDataReader reader)
    {
        Project project = new Project
        {
            Id = (Guid) reader.GetValue(0),
            Name = (string) reader.GetValue(1),
            Description = (string) reader.GetValue(2),
            User = UserGetById((Guid)reader.GetValue(3))
        };

        projects.Add(project);
    });

    return projects;
}
```

Na předešlý kód jsou navázány další metody, které vracejí buď celou kolekci, nebo pouze jeden prvek. Pro zjednodušení zde uvádím pouze příklady těchto metod, jak by mohli být implementované.

Tato metoda získává z ProjectReaderu generickou kolekci IList. Díky podpoře LINQ upravuje vlastní návratovou hodnotu pouze na jediný prvek z této kolekce, protože výsledek pro daný Sql dotaz vrací vždy pouze jeden záznam.

```
public Project ProjectGetById(Guid id)
{
    SqlCeCommand command =
    new SqlCeCommand
    ("SELECT Id, Name, Description FROM Project WHERE Id = @Id");

    command.Parameters.AddWithValue("@Id", id);
    return ProjectReader(command).FirstOrDefault();
}
```

Druhá metoda vrací pouze výsledek ProjectReaderu, protože vrací stejný typ, jako je návratový typ metody ProjectGet All.

```
public IList<Project> ProjectGetByUserId(Guid id)
{
    SqlCeCommand command =
    new SqlCeCommand("SELECT Id, Name, Description, UserId FROM Project);
    return ProjectReader(command);
}
```

15 Testování

Jedná se o nedílnou součást softwarového vývoje a je mu přikládána stále větší důležitost. Poskytuje tvůrcům aplikací kontrolu kvality výsledného produktu. Testování je poměrně široký pojem, který se dělí na několik typů podle fáze životního cyklu vývoje. Každý typ je možno dále rozdělit na několik typů, resp. kategorií. Při vývoji aplikace jsem využil dva typy, první je testování zdrojového kódu pomocí nástrojů k tomu určených a druhým je uživatelské testování neboli testování za pomocí reálných uživatelů dané aplikace.

15.1 Testování zdrojového kódu

Toto by mělo být zvykem každého dobrého programátora, protože mu usnadňuje práci a ověření správné funkčnosti, zejména když vytváří knihovny nebo aplikace, které nemají uživatelské rozhraní. Právě zde je testování neocenitelným pomocníkem, protože se stává jedinou metodou jak ověřit správnou funkčnost knihovny nebo aplikace. Dalším případem je, když vývojář provádí refaktoring, který je metodou pro zlepšení kvality zdrojového kódu.

Existuje i metodika vývoje software TDD (Test-Driven Development, česky programování řízené testy), která zjednodušeně říká, že první co by programátor měl udělat je napsat test a tento test by měl dopadnout neúspěšně. Dalším krokem je teprve napsání vlastní implementace testované části, která splní test. A tento cyklus stále opakovat, dokud test nedopadne úspěšně. Pokud test projde, měla by následovat refaktorizace kódu za zlepšením kvality kódu. [8]

Nejkritičtějším místem každé aplikace pracující s daty je právě jejich samotné ukládání a načítání. Proto jsem se zaměřil na testování právě této části mé aplikace. Snahou bylo otestovat správnou funkčnost vrstvy, která se stará o práci s databází. Konkrétně se jedná o objekt DataLayerImpl, který implementuje rozhraní IDataLayer.

Pro testování jsem využil open-source framework pro unit testy NUnit. Jedná se o port JUnit frameworku, známého z prostředí Javy. NUnit je určen pro .NET Framework, je kompletně napsaný v jazyce C#.

Co musí předcházet samotnému psaní testů je uvedení databáze do známého stavu. Jedná se o vymazání všech dat z databáze a uložení předem vytvořených pro účely testování, aby byly záznamy předem určené a tudíž snadno testovatelné. Pro vytvoření výchozího stavu se využívá metoda s atributem TestFixtureSetUp, která se spustí při spuštění jakéhokoliv testu z dané třídy s testy. Metoda CreateInitialData obsahuje vložení předem připravených testovacích dat, které se poté využívají v jednotlivých testech.

```
[TestFixtureSetUp]
public void TestFixtureSetUp()
{
    CreateInitialData();
}
```

Pro ukázku zde uvádím test pro ověření správného načtení objektu projekt z vrstvy pro práce s databází. Při psaní testů mě nezajímá, jak pracuje volaná metoda uvnitř, z mého pohledu se jedná o černou skříňku. Pouze jí zadám požadované vstupní parametry a ověřuji správnost vráceného výsledku. Atribut Test označuje, že daná metoda je testem a NUnit framework jí umí zpracovat a vyhodnotit jestli test prošel úspěšně nebo neúspěšně. Metoda ProjectGetId očekává parametr typu Guid, který je Id daného projektu. Tento identifikátor jsem si k projektu přiřadil při vytváření testovacích dat a inicializaci výchozího stavu databáze před spuštěním samotného testu. Následně ověřuji, jestli se vrátil očekávaný objekt, dle správného Id, jména a popisu projektu.

```
[Test]
public void ProjectGetByIdTest()
{
    Guid projectId = new Guid("DA35983B-6524-4DA7-ABB3-4C0AB88EAAD2");

    Project result = _dataLayer.ProjectGetById(projectId);

    Assert.AreEqual(projectId, result.Id);
    Assert.AreEqual("Project 1", result.Name);
    Assert.AreEqual("Project One", result.Description);
}
```

15.2 Testování aplikace v emulátoru

Jedná se o testování v softwarově emulovaném prostředí Windows Mobile, které běží jako aplikace na počítači programátora aplikace. Toto prostředí se chová jako klasické zařízení, které by bylo k počítači připojené například pomocí USB. Také obsahuje plnohodnotný systém Windows Mobile.

Výhodou je také provázanost s vývojovým prostředím Visual Studio, díky které se programovaná aplikace nahrává přímo do emulátoru a je možno ji bez problému ladit. Samotný emulátor je distribuován právě s Visual Studiem 2008, případně je možno si ho stáhnout samostatně.[21] Microsoft Device Emulátor není určen pouze pro simulaci zařízení Windows Mobile, ale umožňuje provozovat veškeré nadstavby nad Windows CE. Jsou to například Pocket PC 2003, Windows CE, Windows Mobile 5 nebo Windows Mobile 6.

Další předností je právě podpora více prostředí, kdy není vývojář nucen vlastnit velké množství fyzických zařízení, ale může si je jednoduše emulovat v tomto prostředí a následně v nich testovat svojí aplikaci.

V práci jsem emulátor využíval zejména k ladění uživatelského prostředí a jeho správné funkčnosti.

15.2.1 Fake GPS

Pro testování aplikací, které vyžadují GPS přijímač, poskytuje Microsoft aplikaci Fake GPS. Tento software simuluje reálný provoz a uživatel si může definovat i jaká data bude tato aplikace poskytovat jako GPS souřadnice. Jedná se o jednoduchý textový soubor, kde jsou v NMEA formátu uloženy jednotlivé souřadnice. [18]

16 Uživatelské rozhraní

Mobilní zařízení jsou specifické právě uživatelským rozhraním. Neexistuje zde žádná klávesnice a myš, tak jak je známe z běžných počítačů. Proto je třeba na toto při tvorbě aplikace myslet a před samotným počátkem vývoje vyhodnotit pro jaký typ zařízení bude výsledek určen.

Tvorbu uživatelského rozhraní ovlivňují zejména dva faktory. Prvním je výkon zařízení, který může ovlivnit vzhled a možnosti využití náročných grafických prvků, případně nebude poskytovat dostatečný výkon pro provoz novější verze operačního systému, který může podporovat lepší zobrazení ovládacích prvků aplikace. Druhým parametrem je rozlišení displeje. Toto je asi nejvíce bolestný parametr při vývoji aplikace, protože existují desítky velikostí a rozlišení displejů, resp. zařízení, na kterých lze provozovat Windows Mobile.

Pokud je aplikace zaměřena pro masové používání, měl by se vývojář zaměřit a zařízení, které mají největší podíl na trhu a pro každé zvlášť přizpůsobit svojí aplikaci. Jedná se právě o uživatelské rozhraní, které by mělo být pro každý typ zařízení a každý druh rozlišení jedinečné. Je zde i další překážka při tvorbě aplikace a tím je i omezená plocha určené samotné aplikaci. Toto by bylo nad rozsah této práce, proto se v ní tímto nezabývám a aplikace je tvořena jedním typem uživatelského rozhraní.

Důležitá je zejména intuitivnost aplikace, aby uživatel neměl problém s pochopením ovládání a i bez uživatelského manuálu byl schopný s aplikací pracovat.

V dnešní době je kladen důraz na dotykové ovládání, tento trend započal s příchodem zařízení iPhone od společnosti Apple. Což sebou přináší další problém, že na už tak malém displeji je potřeba, aby ovládací prvky byli dostatečné veliké pro ovládání dotykem. Což sebou přináší větší nepřesnost oproti ovládání pomocí dotykového pera.

17 Uživatelský popis aplikace

17.1 První spuštění

Obrázek 17-1 - Úvodní obrazovka

Po spuštění aplikace se objeví úvodní obrazovka s formulářem pro přihlášení uživatele. Jelikož ještě žádný uživatel neexistuje, je potřeba stisknout tlačítko „New User“ (Nový uživatel), pro vytvoření nového uživatele.

Obrázek 17-2 - Nový uživatel

Otevře se formulář pro zadání údajů o novém uživateli. Jsou to Name (Jméno), Username (Uživatelské jméno), které slouží pro přihlášení do aplikace a Password (Heslo), jako ověřovací mechanismus.

Obrázek 17-3 - Nový uživatel po validaci

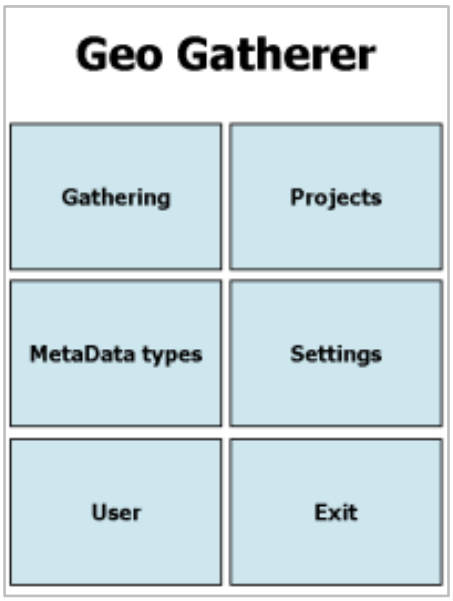
Po zadání nových údajů je potřeba je uložit stiskem tlačítka Save. Zároveň je nový uživatel automaticky přihlášen do aplikace.

17.2 Přihlášení

Obrázek 17-4 – První přihlášení

Pro přihlášení do aplikace stačí vyplnit úvodní formulář uživatelským jménem a heslem. Následně stisknout tlačítko Login (přihlášení).

17.3 Hlavní menu



Obrázek 17-5 – Hlavní menu

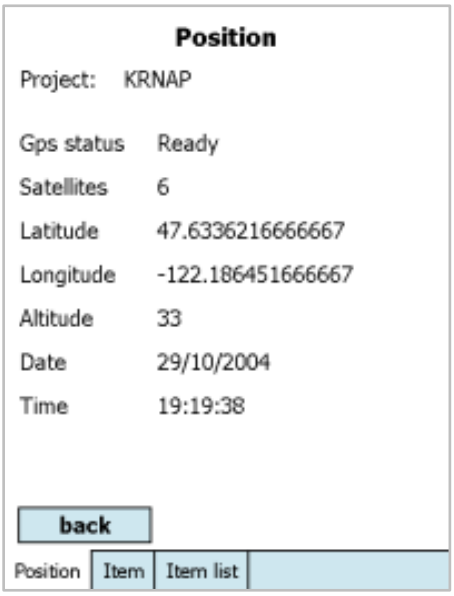
Hlavní menu aplikace poskytuje základní rozcestník pro práci s aplikací.

- Gathering – sběr dat
- Projects – prohlížení uživatelových projektů
- MetaData types – definování projektových datových typu
- Settings – nastavení aplikace
- User – nastavení uživatele a změna hesla
- Exit – ukončení aplikace

17.4 Sběr dat

Obrázek 17-6 – Výběr projektu pro sběr

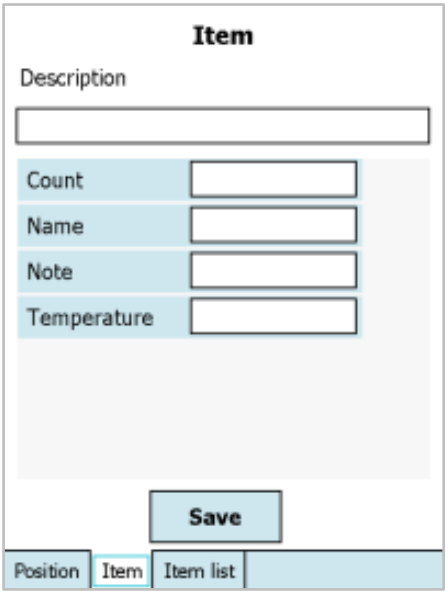
První co je potřeba před začátkem sběru dat, je vybrat projekt ke kterému se mají data ukládat. Uživatel vybere projekt z rozevíracího menu a stiskne tlačítko „Gathering“ pro spuštění sběru dat.



Obrázek 17-7 – Informace o aktuální poloze

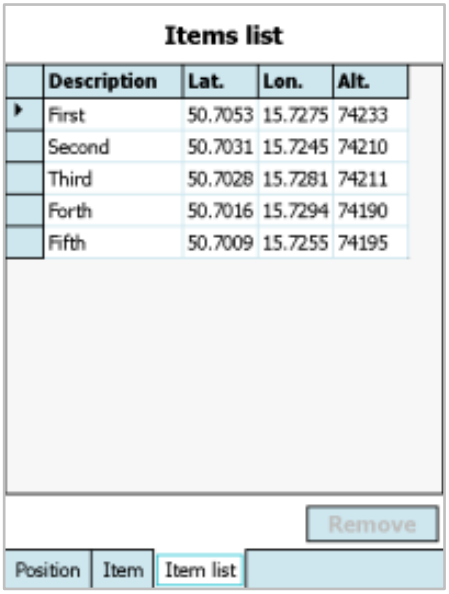
Obrazovka Position (poloha) zobrazuje stav GPSky a aktuální údaje z ní, jako je počet satelitů, zeměpisná šířka, zeměpisná délka, nadmořská výška nebo přesný čas. Zároveň je zobrazen i název projektu, ke kterému je možné sbírat data.

Ve spodní části jsou záložky pro přecházení mezí jednotlivými obrazovkami sbírání dat.



Obrázek 17-8 – Formulář pro novou položku

Obrazovka Item (položka) je určena pro vytvoření nového záznamu s určenou zeměpisnou polohou. Je zde položka pro krátký popis daného záznamu a zobrazení položek pro uložení metadat k dané poloze. Tlačítkem Save (uložení) se data z formuláře uloží do databáze a přiřadí se k nim aktuální poloha z GPS přijímače.



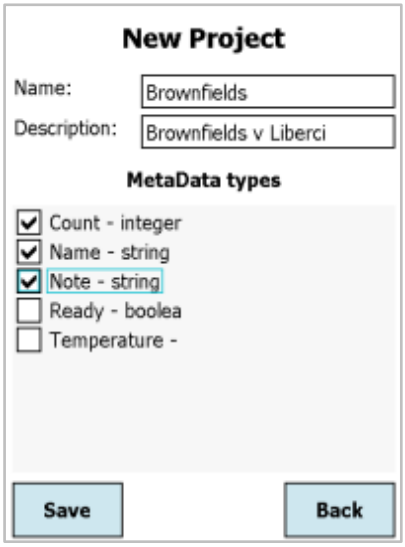
Obrázek 17-9 –Seznam záznamů u projektu

Poslední záložka „Item list“ slouží pro zobrazení všech záznamů k aktuálnímu projektu. Pomocí tlačítka „remove“, lze záznamy mazat. Například, pokud uživatel uložil položku na špatném místě.

17.5 Projekty

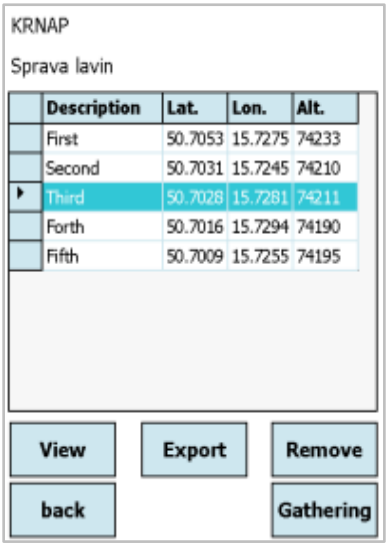
Obrázek 17-10 – Seznam projektů

Seznam projektů je rozcestníkem pro několik možností. Je zde volba pro vytvoření nového projektu „New“, pro otevření stávajícího „Open“ nebo jeho smazání „Delete“.



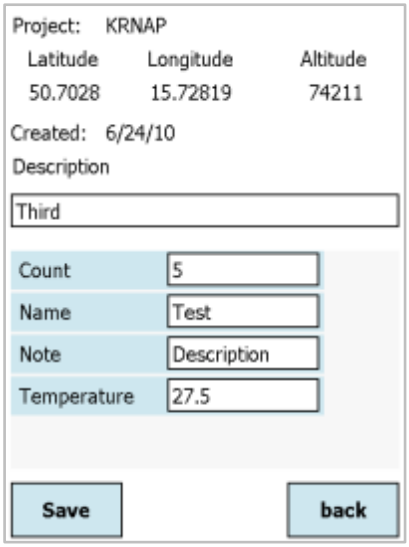
Obrázek 17-11 – Založení nového projektu

Založení nové projektu je velice snadné. Stačí zadat jeho jméno (Name), krátký popisek (Description) a vybrat, která data chceme k novému projektu sledovat, pomocí zaškrtávacích polí v oblasti „MetaData types“. Následné stisknutí tlačítka „Save“ tento projekt uloží.



Obrázek 17-12 – Seznam záznamů u projektu

Na obrazovce detailu projektu, je možné vidět přehled jednotlivých záznamů a případně si je vyexportovat stisknutím tlačítka „Export“. Případně nežádoucí položky vymazat pomocí „Remove“. Je zde i možnost přímo z projektu přejít do sběru dat a to tlačítkem „Gathering“.



Obrázek 17-12 – Detail záznamu

Detail záznamu uživateli poskytuje zobrazení zaznamenaných hodnot a jejich případnou zpětnou editaci.

17.6 MetaData typy

Obrázek 17-13 – Editace MetaData typů

Formulář „MetaData types – settings“ uživateli umožňuje přidávat, editovat a mazat datové typy v rámci aplikace.

U nového typu jsou k dispozici čtyři datové typy.

- string – textové hodnoty
- integer – celočíselné hodnoty
- float – hodnoty s desetinou čárkou
- bool – pro logické hodnoty

17.7 Nastavení

Obrázek 17-14 - Nastavení

V obrazovce nastavení je možno změnit dobu, jak často se mají získávat hodnoty z GPS přijímače do aplikace. Výchozí hodnota je 1 vteřina.

17.8 Uživatelé

Obrázek 17-15 – Nastavení uživatele

Uživatel má možnost si, po přihlášení do aplikace, změnit své údaje, včetně změny hesla. Tuto změnu potvrzuje tlačítkem „Save“ (Uložit)

18 Možný vývoj do budoucna

Jistě existuje spousta možností, jak by se měla tato aplikace vyvíjet do budoucna a jaké by měla obsahovat prvky. Já zde uvedu několik, které si myslím, že jsou stěžejní a přispějí ke kvalitnější uživatelské přívětivosti a zkušenosti.

18.1 Serverová část

Aplikace díky podpoře více uživatelů je přípravě i proto, aby jí bylo možné propojit se serverovým řešením. To by mělo umožňovat nahrávání dat přímo na server bez nutnosti jejich exportování v aplikaci. Data na serveru by měla být přístupná pomocí webového rozhraní, které by mělo umožňovat si prohlížet uložená data přímo nad mapovými podklady, jako jsou například Google Maps nebo Bing Maps. Zároveň by zde měla být možnost jejich editace a případně zpřesňování polohy, právě nad zmíněnými kartografickými podklady.

18.2 Grafické komponenty

Základní grafické komponenty pro tvorbu GUI v .NET Compact Frameworku jsou značné omezené a rozhodně nejsou přizpůsobeny pro ovládání dotykem prstů. Proto by bylo vhodné nahradit stávající grafické prvky, nějakým lepším řešením. Což díky využitému návrhovému vzoru MVP, kde je samotné zobrazení odděleno od jeho logiky, by neměl být zásadní problém. Ve chvíli psaní této práce jsou k dispozici tři projekty, které poskytují grafické prvky pro Windows Mobile zadarmo. Prvním je SenseSDK [15] za kterým stojí lidé z komunity xda-developers, druhým je projekt Fluid [16], který je umístěn na Codeplex.com a poslední je Silvermoon [17], což je knihovna grafických komponent, které pro vykreslování používají OpenGL.

18.3 Lokalizace

Nyní je celé uživatelské rozhraní v angličtině. Nejedná se o žádné složité gramatické jevy, proto i bez znalostí tohoto jazyka by uživatel neměl mít problém s užíváním aplikace. Ovšem bylo by dobré do budoucna vytvořit více lokalizací např. německou, českou nebo francouzskou.

19 Případy užití

Aplikace je vhodná pro všechny scénáře, kde je potřeba sběr informací v terénu a hodnota těchto informací se zvyšuje se znalostí jejich přesné polohy. Pro tuto aplikaci existuje několik možných případů využití v praxi, ať již komerčními subjekty nebo veřejnými, jako jsou obecní úřady, technické služby nebo správa komunikací.

19.1 Městská policie

Tato aplikace najde uplatnění například u městské policie, která pomocí ní může sbírat data o špatně stojících vozech, kterým je nasazena botička. Strážníci mohou sledovat údaje, jako jsou SPZ, výrobce vozu, typ vozu a další a tyto informace poté vyhodnocovat, což záleží na systému třetích stran, ve kterém jsou informace z této aplikace použity. Například jaké značky vozů nejčastěji dostávají botičky za špatné parkování a v jakém místě.

19.2 Studenti geografie

Aplikace je velice vhodná i pro studenty vysokých škol, kteří studují obor geografie, resp. předměty související. V aplikaci si pro každý semestrální projekt mohou definovat sledované metadata, která pak v terénu jednoduše pomocí svého Windows Mobile zařízení sledují. Díky podpoře více uživatelů aplikace na jednom zařízení je zde i možnost, sdílení zařízení mezi studenty. Toto zařízení může být například zapůjčeno od školy. Tyto data si mohou vyexportovat a využít je například v programu ArcGIS od společnosti ESRI, který jim umožní provádět analýzy nad sesbíranými daty.

Zejména se jedná o jevy, ke kterým neexistující žádné primární zdroje dat. Například semestrální práce na téma Kámen v krajině může být typickým příkladem takovéhoto projektu. Student má za úkol zmapovat výskyt kamenů resp. skalních útvarů na daném území. K tomuto tématu existují mapy, ale ty obsahují pouze významné prvky, tudíž ne všechny. Student si nadefinuje, jaká data chce sledovat u konkrétních výskytů, mohou to být typy skalních útvarů, jejich původ, typ horniny a řada dalších parametrů, které využije ve svém semestrálním projektu. Bez této aplikace by musel využít klasickou mapu, papír a tužku. Bohužel toto příslušenství podléhá rychlé zkáze a není tak dobře použitelné v terénu, jako zařízení s aplikací, kde je vše připraveno pro rychlé zadání údajů. Po návratu z venkovního prostředí si student vyexportuje data a v klidu domova je může zpracovávat pro další využití.

20 Závěr

Mým hlavním cílem bakalářské práce bylo naprogramovat aplikaci pro sběr dat s určenou zeměpisnou polohou na platformě Windows Mobile. Díky této práci jsem si osvojil znalosti vývoje pro mobilní zařízení s tímto operačním systémem. Dále jsem prozkoumal i další platformy a jejich možnosti vývoje aplikací.

Splnil jsem předem vytýčený cíl, kterým bylo naprogramovat aplikaci, která by uživatelům poskytovala jednoduchý způsob, jak zaznamenávat data společně s polohou. Díky této práci jsem získal znalosti problematiky tvorby aplikací pro mobilní zařízení. Osvojil jsem si technologie, které jsou k dispozici na platformě Windows Mobile, zejména omezení SQL CE oproti běžným SQL serverům. Dále jsem pronikl hlouběji do fungování globálních polohových systému a získal znalosti o jejich architektuře a aktuálním stavu.

Při programování pro mě bylo zejména zajímavé vyřešení vlastního jednoduchého O/R mapperu, protože pro .NET Compact Framework není k dispozici žádný bezplatný. Další zajímavou zkušeností byla tvorba uživatelského rozhraní, protože na mobilním zařízení je člověk značně omezen velikostí plochy.

Pokračování této práce jsem nastínil v kapitole 18, kde jsem se snažil ukázat hlavní nedostatky aplikace do budoucna a případně určit její další směřování. Zejména rozšíření o spolupráci se serverovým řešením by mohlo být velice zajímavé z pohledu implementace. Tato část sebou přináší novou problematiku, jako jsou komunikace po GSM síti, tvorba webového rozhraní nebo spolupráce s mapovými podklady mapových serverů.

21 Použité zdroje a literatura

[1] Application Development in the .NET Compact Framework [online]
URL: <http://msdn.microsoft.com/en-us/library/ms172489(v=VS.90).aspx>
[cit. 2010-07-10]

[2] Baijian Yang, Pei Zheng, Lionel M. Ni. Professional Microsoft Smartphone Programming. 1. vydání. Wrox, 2007. 494 s. ISBN 978-0471762935

[3] C# Programming Guide [online]
URL: <http://msdn.microsoft.com/en-us/library/67ef8sbd.aspx > [cit. 2010-07-10]

[4] Global Positioning System [online]
URL: <http://www.gps.gov> [cit. 2010-07-10]

[5] Tuček Ján. Geografické informační systémy – Principy a praxe. 1. vydání. Brno: Computer Press, 1998. 424 s. ISBN 80-7226-091-X

[6] Implementing Singleton in C# [online]
URL: <http://msdn.microsoft.com/en-us/library/ms998558.aspx> [cit. 2010-07-10]

[7] Boodhoo Jean-Paul. Model View Presenter [online]
URL: <http://msdn.microsoft.com/en-us/magazine/cc188690.aspx> [cit. 2010-07-10]

[8] Kadlec Václav. Agilní programování – Metodiky efektivního vývoje software Computer Press, 2004. 280 s. ISBN 80-251-0342-0

[9] Meier J.D. a kolektiv autorů. Mobile Application Architecture Guide [online]
URL: <http://apparch.codeplex.com/releases/view/19798> [cit. 2010-07-10]

[10] Overview of the .NET Compact Framework [online]
URL: <http://msdn.microsoft.com/en-us/library/w6ah6cw1(v=VS.90).aspx>
[cit. 2010-07-10]

[11] Rapant Petr. Družicové polohové systémy. 1. vydání Ostrava: Vysoká škola báňská – Technická univerzita Ostrava, 2002. 144 s. ISBN 80-248-0124-8

[12] Rapant Petr. Geoinformační technologie. 2. Vydání Ostrava: Vysoká škola báňská – Technická univerzita Ostrava, 2006. 126 s. ISBN 80-248-1263-0

[13] Robinson Simon a kolektiv autorů. C# Programujeme profesionálně. 1. vydání. Brno: Computer Press, 2003. 1130 s. ISBN 80-251-0085-5. Překlad: Bogdan Kiszka

[14] Overview of SQL Server Compact Edition [online]
URL: <http://msdn.microsoft.com/en-us/library/ms172448(v=SQL.90).aspx>
[cit. 2010-07-10]

[15] Eboelzner. Sense Interface SDK
Knihovna s grafickými komponentami pro Windows Mobile
URL: <http://forum.xda-developers.com/showthread.php?t=648906> [cit. 2010-07-10]

[16] Thomas Gerber. Fluid – Windows Mobile .NET Touch Controls
Knihovna s grafickými komponentami pro Windows Mobile
URL: <http://fluid.codeplex.com> [cit. 2010-07-10]

[17] Thomas Gerber. Silvermoon
Knihovna s grafickými komponentami pro Windows Mobile
URL: <http://silvermoon.codeplex.com> [cit. 2010-07-10]

[18] Using the FakeGPS Utility
URL: <http://msdn.microsoft.com/en-us/library/bb158722.aspx> [cit. 2010-07-10]

[19] Wigley Andy,Moth Daniel, Foot Peter. Microsoft Mobile Development Handbook. 2. vydání. Microsoft Press, 2007. 688 s. ISBN 978-0735623583

[20] Heil Henry. The History of Windows Mobile [online]
URL: <http://www.brighthub.com/computing/windows-platform/articles/1295.aspx> [cit. 2010-07-10]

[21] Microsoft Device Emulator [online]
Emulátor zařízení s operačním systémem Windows CE
URL: <http://www.microsoft.com/downloads/details.aspx?familyid=a6f6adaf-12e3-4b2f-a394-356e2c2fb114&displaylang=en> [cit. 2010-07-10]

22 Seznam zkratek

ADO.NET	Active Data Objects for .NET - Knihovny pro objektovou práci s relačními daty na nejnižší úrovni
AJAX	Asynchronous JavaScript and XML - Technologie pro změnu obsahu webových stránek bez nutnosti jejich znovunačtení
API	Application Programmin Interface – Rozhraní pro programování
ČR	Česká republika
GPS	Global Positioning System – Družicový navigační systém provozovaný Ministerstvem obrany Spojených států amerických
GIS	Geographic information system – Geografický informační systém
GSM	Global System for Mobile Communications – Globální systém pro mobilní komunikaci – standart pro mobilní telefony
GUI	Graphical User Interface – Grafické uživatelské rozhraní
HTML5	HyperText Markup Language – Značkovací jazyk pro hypertext ve verzi 5
IDE	Integrated Development Environment – Vývojové prostředí
IrDA	Infračervený port standardizovaný konsorciem Infrared Data Association
JIT kompilace	Just In Time kompilace – Překlad za běhu programu
LINQ	Language Integrated Query – Jazyk pro dotazování nad jakýmikoliv daty
MVC	Model-View-Controller – Architektonický vzor pro uživatelské rozhraní
MVP	Model-View-Presenter – Odnož MVC vzoru
NATO	North Atlantic Treaty Organization – Severoatlantická aliance
ORM	Object-relational mapping – Technika pro konverzi dat mezi dvěma nekompatibilními typovými systémy
PDA	Personal Digital Assistant – Osobní digitální pomocník neboli kapesní počítač
POCO	Plain Old CLR Object – Objekty bez vnitřní logiky
RAM	Random-Access Memory – Paměť s přímým přístupem

RIA	Rich Internet application – Moderní webové aplikace zaměřené na kvalitu ovládání a lepší interakci s uživatelem
ROM	Read-Only Memory – Paměť určená pouze pro čtení
SDF	SQL Server Compact Edition Database File – formát souboru pro uložení databáze
SPZ	Státní poznávací značka
S-42	Mapový systém
S-JTSK	Souřadný systém jednotné trigonometrické sítě katastrální
TDD	Test-driven development – Programování řízené testy
USA	United States of America – Spojené státy americké
USB	Universal Serial Bus – Univerzální sériová sběrnice
VoIP	Voice over Internet Protocol – Technologie pro přenos digitalizovaného hlasu pomocí protokolů UDP/TCP/IP
WiFi	Standard pro lokální bezdrátové sítě
WGS-84	World Geodetic System 1984 – Světový geodetický systém 1984
XML	Extensible Markup Language – Obecný značkovací jazyk
.NET CF	.NET Compact Framework

23 Seznam obrázků

24 Přílohy

24.1 Příloha A: Přehled mobilních platforem

Android

Vyvíjí: Open Handset Alliance (65 společností z oblasti mobilních zařízení)

Datum vzniku: 5. listopadu 2007

Založeno na: Linuxu

Programovací jazyky: Java a Python

BlackBerry

Vyvíjí: Research In Motion

Datum vzniku: 1999

Programovací jazyky: Java

Symbian

Vyvíjí: Symbian Foundation

Datum vzniku: červen 2001

Programovací jazyky: C++

iPhone

Vyvíjí: Apple

Datum vzniku: 29. června 2007

Založeno na: Mac OS X (Darwin s jádren XNU)

Programovací jazyky: Objective-C

WebOS

Vyvíjí: Hawlett-Packard

Datum vzniku: 8. leden 2009

Založeno na: Linuxu

Programovací jazyky: JavaScript

Windows Mobile

Vyvíjí: Microsoft

Datum vzniku: 19. květen 2000

Založeno na: Windows CE

Programovací jazyky: C++, C#, VB.NET

Bada

Vyvíjí: Samsung Electronics

Datum vzniku: 10. listopad 2009

Založeno na: Linuxu

Programovací jazyky: C++

24.2 Příloha B: Přehled družicových polohových systémů

Galileo

Země: Evropská unie

Stav: V realizaci

Počet družic: 30 (plán)

GPS

Země: Spojené státy americké

Stav: V provozu

Počet družic: 30

GLONASS

Země: Rusko

Stav: V provozu

Počet družic: 21

Compass

Země: Čína

Stav: V realizaci

Počet družic: 35 (plán)

Beidou

Země: Čína

Stav: V provozu

Počet družic: 4

DORIS

Země: Francie

Stav: V provozu

Počet družic: 0 (pozemní lokalizace)

IRNSS

Země: Indie

Stav: V realizaci

Počet družic: neznámý

QZSS

Země: Japonsko

Stav: V realizaci

Počet družic: neznámý